Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГТУ»)**

**Институт компьютерных систем и информационной безопасности**

**Кафедра информационных систем и программирования**

**ОТЧЁТ**

**Дисциплина: Тестирование и отладка программного обеспечения**

**Тема: «Выделение бизнес-логики и применение различных вариантов визуального представления»**

**Лабораторная: 9**

Работу выполнил: Басма Нассиф Хуссейн

Направление подготовки: 09.03.04 Программная инженерия

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Степанова

**Цель работы:** изучить техники рефакторинга и способы совместного использования общего кода в различных приложениях.

**Контрольные вопросы**

1. **Что такое сборка?**

Сборка в контексте .NET является компилированным кодом, который исполняется средой CLR (Common Language Runtime). Она содержит один или несколько файлов, которые могут включать управляемый код (.dll или .exe), ресурсы, типы данных и метаданные, описывающие содержимое сборки. Сборки являются основным строительным блоком приложений .NET и служат единицей развертывания, версионности и безопасности.

1. **Как создаются динамические библиотеки?**

Динамические библиотеки (DLL - Dynamic Link Libraries) в .NET создаются путем компиляции кода с использованием компилятора языка, такого как csc.exe для C# или vbc.exe для VB.NET, с указанием ключа компиляции для создания библиотеки (например, /target:library для csc.exe). В проектах Visual Studio можно создать проект типа "Class Library", который автоматически настроен на создание DLL. После компиляции проекта генерируется файл DLL, который содержит скомпилированный код и может быть использован в других приложениях.

1. **Как связать и использовать библиотеку в своем приложении?**

Для использования динамической библиотеки в своем приложении необходимо сначала добавить ссылку на нее. В Visual Studio это делается путем добавления ссылки на сборку в проект через контекстное меню "Ссылки" (References) в обозревателе решений. После добавления ссылки на DLL можно использовать ее типы, методы и свойства в коде, добавив соответствующий оператор using (C#) или Imports (VB.NET) с именем пространства имен библиотеки. Компилятор учитывает информацию о сборке при сборке приложения, а CLR загружает и связывает сборку во время выполнения.

**Задание**

**1) Создать новое оконное приложение и библиотеку классов в соответствии с предложенными в пункте 2.4 изменениями, применив описанные в теории методы рефакторинга.**

1. **Создать новый проект C# Class Library.**

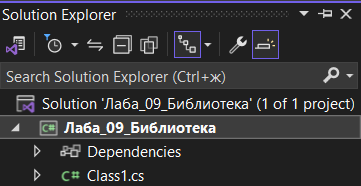


Рисунок 1 – Создание библиотеки

1. **Перенести в него основные классы старого консольного приложения (все кроме Program).**

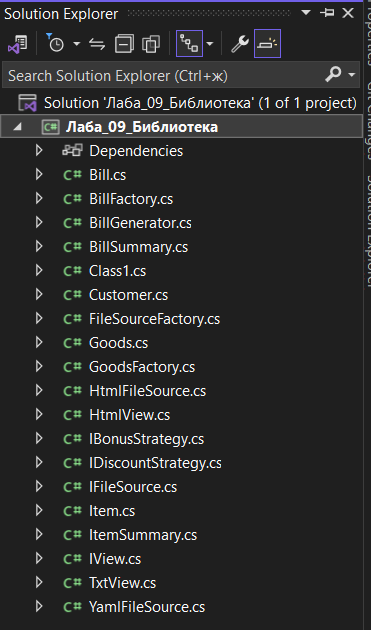


Рисунок 2 – Перенос классов

1. **Добавить ссылку на новую библиотеку в консольном приложении.**

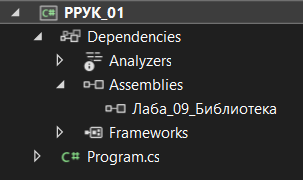


Рисунок 3 – Добавление ссылки на новую библиотеку

1. **Добавить пространство имен новой библиотеки в консольное приложение.**

using System.Text;

namespace РРУК\_01

{

public class Program

{

static void Main(string[] args)

{

string filename = "BillInfo.yaml";

if (args.Length == 1)

filename = args[0];

IFileSource fileSource = FileSourceFactory.CreateFileSource(filename);

using (FileStream fs = new FileStream(filename, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

BillFactory factory = new BillFactory(fileSource);

BillGenerator bill = factory.CreateBill(sr, "NewYearsSettings.json");

string billOutput = bill.GetBIll();

Console.WriteLine(billOutput);

}

}

}

}

1. **Обновить ссылки на тестируемые классы в тестовом проекте.**

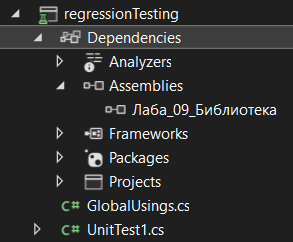


Рисунок 4 – Обновленная ссылка

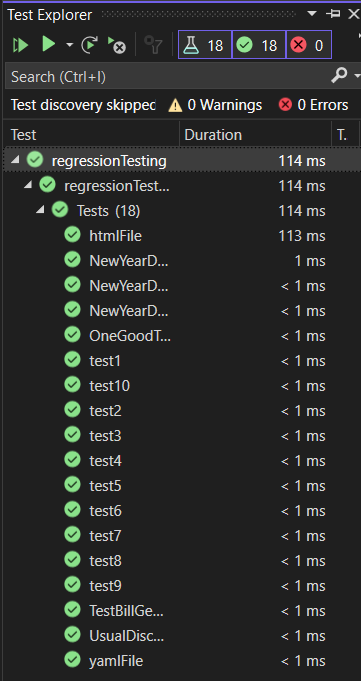


Рисунок 5 – Результат тестирование после обновление ссылки

1. **Создать новый проект оконного приложения C# Windows Forms Application.**

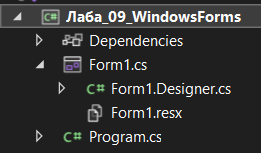


Рисунок 6 – Создание Windows Forms

1. **Добавить ссылку на новую библиотеку в оконном приложении.**

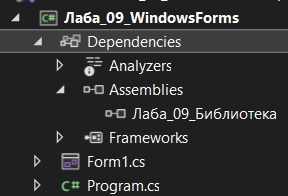


Рисунок 7 – Добавление ссылки в Windows Forms

1. **Добавить пространство имен новой библиотеки в оконное приложение.**

namespace Лаба\_09\_WindowsForms

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

}

}

1. **Добавить необходимые визуальные компоненты (диалоги для выбора загружаемого файла и результатов работы, поле для вывода результатов генерации чека, кнопки управления работой (запуск генерации, выход и т.д.), переключатели для выбора формата вывода (HTML/TXT) и т.д.)**

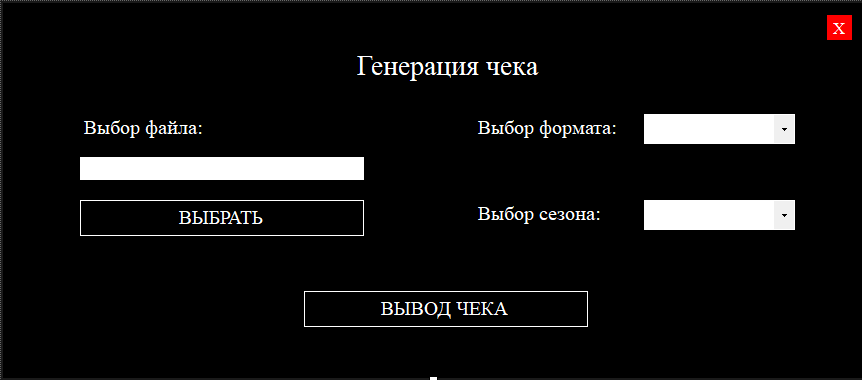


Рисунок 8 – Визуализация

1. **Связать визуальные компоненты с кодом библиотеки.**

// Form1

using System.Windows.Forms;

using РРУК\_01;

namespace Лаба\_09\_WindowsForms

{

public partial class Form1 : Form

{

string fileName;

string seasonName;

string billResult;

IView view;

ViewForm viewForm;

public Form1()

{

InitializeComponent();

cbFormat.SelectedIndex = 0;

cbSeason.SelectedIndex = 0;

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnChoose\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

InitialDirectory = @"C:\Users\basma\Desktop\3 year\6 semester\Рефакторинг и работа с унаследованным

кодом\Лабы\Лаба\_09\Лаба\_09\_WindowsForms\bin\Debug\net8.0-windows",

Filter = "Выберите файл: (\*.yaml;\*.html;.htm)|\*.yaml;\*.html;\*.htm|" +

"HTML файл: (\*.html;\*.htm)|\*.html;\*.htm|" +

"Yaml файл: (\*.yaml)|\*.yaml",

FilterIndex = 0,

RestoreDirectory = true,

};

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

string filePath = openFileDialog.FileName;

fileName = Path.GetFileName(filePath);

tbFileName.Text = fileName;

}

}

private void btnRun\_Click(object sender, EventArgs e)

{

ChooseFortmat();

ChooseSeason();

if (fileName != null && seasonName != null)

{

IFileSource fileSource = FileSourceFactory.CreateFileSource(fileName);

using (FileStream fs = new FileStream(fileName, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

if (view != null)

{

BillFactory factory = new BillFactory(fileSource);

BillGenerator bill = factory.CreateBill(sr, seasonName, view);

billResult = bill.GetBIll();

if (view.GetType() == typeof(HtmlView))

{

viewForm = new ViewForm(billResult, "html");

viewForm.Show();

}

else

{

viewForm = new ViewForm(billResult, "txt");

viewForm.Show();

}

}

}

}

}

private void ChooseFortmat()

{

if (cbFormat.SelectedIndex == 0)

{

// txt

view = new TxtView();

}

else if (cbFormat.SelectedIndex == 1)

{

// html

view = new HtmlView();

}

}

private void ChooseSeason()

{

if (cbSeason.SelectedIndex == 0)

{

// normal

seasonName = "RegularSettings.json";

}

else if (cbSeason.SelectedIndex == 1)

{

// new year

seasonName = "NewYearsSettings.json";

}

}

}

}

// ViewForm

namespace Лаба\_09\_WindowsForms

{

public partial class ViewForm : Form

{

private RichTextBox richTextBox;

private WebBrowser webBrowser;

public ViewForm(string content, string choice)

{

InitializeComponent();

if (choice == "txt")

{

richTextBox = new RichTextBox

{

Dock = DockStyle.Fill,

ReadOnly = true,

};

Controls.Add(richTextBox);

richTextBox.Text = content;

}

else if (choice == "html")

{

webBrowser = new WebBrowser();

webBrowser.Dock = DockStyle.Fill;

Controls.Add(webBrowser);

webBrowser.DocumentText = content;

}

}

}

}

**Заключение:**

Были изучены техники рефакторинга и способы совместного использования общего кода в различных приложениях.